Simple PHP Server

- Runs a web server locally on your machine
- Uses built-in PHP web server
- Prints "Hello World" when accessed

PHP Script Example

```php
<?php
// Simple script to print Hello World
 echo 'Hello World';
?>
```

- PHP programs are enclosed by `<?php ... ?>`.
- We can run PHP as a script, such as Python or Node (JavaScript).
- To use this code as a script, run `php index.php`.
- However, we do not use PHP this way.

- To use this code as a server-side code, run `php -S localhost:8000`.
- PHP code inside `<?php ... ?>` is **executed on the server**.
- Server (PHP) translates the PHP code and sends only the **output** (HTML/text) to the browser
- Browser never sees the PHP code, only the final rendered page.
- PHP lets HTML pages include **dynamic content** (e.g., user/session data).

Starting PHP Server

1. Place your PHP file (`index.php`) in a project folder
2. Open a terminal and navigate to this folder
3. Run the server with the command:
   - You need to change the port number (8000) if another process uses it.

```
php -S localhost:8000
```

Accessing the server

- Access http://localhost:8000 via browser
    - You need to change the port number (8000) if you are using a different port.
    - You should see "Hello World" in your browser.

- Access via curl (command line tool)
    - You can also access your PHP server using the curl command, which is helpful for testing from the terminal or scripting API requests.
    - The response (e.g., "Hello World") will be printed directly in your terminal.

```
curl http://localhost:8000
```

Running PHP source file

- When we run php this way, `php` reads the file `index.php` and executes it.
- This is similar to run Python script using Python interpreter.

```
php index.php
```

Using VSCode Extension

- Install "PHP Server" extension (Use **brapifra.phpserver** in the Extension search).
- Open your project folder and choose your PHP file (`index.php`) in the project folder.
- Use the extension:
    - Clicking on VSCode's PHP button (icon on the top-right corner).
    - To stop the PHP server, use the **PHP Server: Stop project** command from the Command Palette.

Debugging a PHP program

- When you need to debug a PHP program, you can choose one of the three choices.
  - Use `error_log() PHP function`: The simplest
  - Use PHPStorm: Simple and Powerful
  - Use Xdebug: Complex, Powerful, and works with VSCode

Use error_log() function

```php
$data = ['foo' => 'bar', 'baz' => 42];
error_log(print_r($data, true), 3, __DIR__ . '/debug.log');
```

Use PHPStorm (Optional)

- Students can request a free license from JetBrains(https://account.jetbrains.com/licenses).
    - Download JetBrains ToolBox App(https://www.jetbrains.com/toolbox-app/)
    - Download PHPStorm
    - Open the PHP Project in PHPStorm.
    - Set breakpoints and run a debugger.

Use Xdebug (Optional)

Check Xdebug installation status

- Command Line

```
php -m | grep xdebug
```

- Web application
  - Make this PHP program, and run the PHP server.
  - We can get all the PHP-related information.
    - Check where the php.ini is located.
    - Check if `Xdebug` is installed

```php
<?php
phpinfo();
?>
```

Installation of Xdebug

- Use https://xdebug.org/docs/install for the installation example.
  - For Mac/Linux, use `pecl`.

```
pecl install xdebug
```

- Check the installation.

```
smcho@m4 ~> php -v
PHP 8.4.11 (cli) (built: Jul 29 2025 15:30:21) (NTS)
Copyright (c) The PHP Group
Built by Homebrew
Zend Engine v4.4.11, Copyright (c) Zend Technologies
    with Xdebug v3.4.5, Copyright (c) 2002-2025, by Derick Rethans
    with Zend OPcache v8.4.11, Copyright (c), by Zend Technologies
```

Update the php.ini

- Find the location of the php.ini file.
- In this example,

```
smcho@m4 ~> php --ini
Configuration File (php.ini) Path: /opt/homebrew/etc/php/8.4
Loaded Configuration File:         /opt/homebrew/etc/php/8.4/php.ini
Scan for additional .ini files in: /opt/homebrew/etc/php/8.4/conf.d
Additional .ini files parsed:      /opt/homebrew/etc/php/8.4/conf.d/ext-opcache.ini
```

- edit "/opt/homebrew/etc/php/8.4/php.ini" to add the following.
- This is an example for Mac.
- For Windows, find the location of the DLL and adjust
- `zend_extension="C:\path\to\php\ext\php_xdebug.dll";`

```
zend_extension="xdebug.so"
xdebug.mode=debug
xdebug.start_with_request=yes
xdebug.client_host=127.0.0.1
xdebug.client_port=9003
```

VSCode

- Install "PHP Debug" extension
- Make ``.vscode/launch.json`

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Listen for Xdebug",
      "type": "php",
      "request": "launch",
      "port": 9003
    }
  ]
}
```

Run XDebug

- In VSCode, in the left activity bar, choose "Run and Debug"
    - Choose "Listen for Xdebug"
- In Terminal, start the server
    - "php -S localhost:8000"
- In the Web browser, access the server
    - Open http://localhost:8000

What You Did

1. You made a web application that prints out "Hello, World" using PHP.
2. You deployed your web application locally on your computer using a PHP web server.
3. You accessed the web application locally via browser.
4. When you deploy your web application using a VPS (Virtual Private Server), anyone can access your web application.

Next Steps

- Connect PHP with MySQL database